IN THE SPECIFICATION

Please amend the Title as follows:

A1 -- A Method for Translating Instructions in a Speculative Microprocessor Featuring Committing State --

Please amend the paragraph at page 3, line 21 as follows:

A2 -- The original method of speculation used by the new processor always ~~updated~~ updates the state of the target processor by committing stores to memory from the gated store buffer and transferring new state to target registers at the end of a sequence of instructions which had executed correctly and before any next sequence of target instructions was translated. This method of updating state is effective for many situations. --

Please amend the paragraph at page 8, line 10 as follows:

A3 -- However, it is possible that in the sequence of target instructions execution of a branch operation will commerce without a commit operation. For example, the translation and optimization software may speculate that one branch will almost always be taken ~~in~~ and consider that direction as the normal direction of the sequence of instructions. In such a case, the software may create a sequence in which the state existing at an exit point is not committed. The sequence to the upper left in the figure beginning at point 10 and providing a branch at point 12 to a second sequence is such a sequence. In such a sequence, no commit operation occurs before the branch because the translation software determines that the sequence from point 10 to point 11 will probably usually be executed and omits the commit in order to accelerate execution. --

Please amend the paragraph at page 9, line 18 as follows:

A4

-- A second problem ~~creating~~ created by the old process is that if a number of branches are taken in a sequence of instructions being executed at points at which target state is uncommitted and a failure occurs during execution at some point after a large number of instructions have been executed, then a great deal of time will have been wasted in executing instructions before rolling back to the last point at which a commit occurred and then retranslating from that point. –

Please amend the paragraph at page 10, line 15 as follows:

A5

-- The present invention obviates these problems of the original process. Figure 2 illustrates a linked set of sequences of translations utilizing the process of the present invention. In this new process, at the beginning of a new sequence before any execution of target instructions has taken place, a commit instruction stores the present state of the operation any memory stores generated by previous correctly-executed sequences of translated instructions. Thus, when ~~a~~ an earlier sequence includes a branch instruction, if that branch is taken to a new sequence of instructions, then the new sequence of instructions begins with a commit operation. If the branch is not taken, the sequence continues to its end at which state of the target processor is known. At this point, execution of a new sequence commences beginning with a commit operation which commits the known state and stores the memory stores generated by execution of the previous sequence. --

Please amend the paragraph at page 11, line 17 as follows:

-- The new process has a number of benefits. First, the length of any sequence of stores which occurs between commit points can only be as long as the longest individual sequence of instructions in a ~~sinlge translations~~ single translation from the initial point at which the last commit occurred (e.g., point 20 to point 21 in Figure 2). Thus the sequence of stores is bounded. This allows close control over the number of uncommitted stores which may reside in the gated store buffer so that overflow of the store buffer is rare. --

Please amend the paragraph at page 11, line 17 as follows:

-- Second, since a commit occurs whenever a branch is taken at the beginning of execution of the next sequence of instructions following the branch, there are no uncommitted branch operations. Thus, if an exception occurs which necessitates a rollback operation after a branch is taken, the amount of work the processor has done which is discarded is limited. For example, the number of stores which must be discarded are limited in ~~number~~ number. Thus ~~orthe~~ the delay attendant on a rollback is significantly reduced. –